Block Cipher and Stream Cipher

February 17, 2009

Internetwork Security - EE548

Symmetric Encryption

- Block ciphers.
- Stream ciphers.
- Extensive work on design of both block and stream cipher.
- Sound theoretical understanding of the primitives: pseudo-random permutation, pseudo-random generators.
- Many practical proposals and their cryptanalysis.

Block Ciphers

- Divide message (plaintext) into fixed size blocks

$$M_1, \ldots, M_i, \ldots$$

- Encrypt each message block separately to obtain $C_i = E_K(M_i)$ where
  - K is the secret symmetric key.
  - $E_K()$ is the encryption function.
- The cipher is $C_1, \ldots, C_i, \ldots$
- Decrypt each cipher block separately to obtain $M_i = D_K(C_i)$ where $D_K()$ is the decryption function.

r-round block cipher

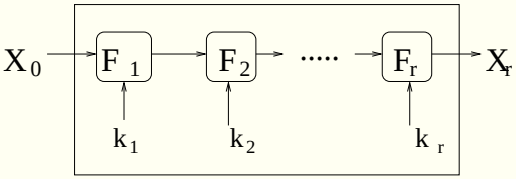

Figure 1: A typical r-round block cipher

- K_1,K_2, ...,K_r is the list of round keys derived from the secret key using a publicly known *key scheduling algorithm.*
- $X_i = F_i(X_{i-1}\ ,\ K_i)$.

Encryption

- Let X be plaintext
- Let F be the round function for all the round
- The encryption operation is carried out as follows:

$$\begin{aligned} X_0 &\leftarrow X \\ X_1 &\leftarrow F(X_0, K_1) \\ X_2 &\leftarrow F(X_1, K_2) \\ &\vdots \\ X_r &\leftarrow F(X_{r-1}, K_r) \end{aligned}$$

- Ciphertext $Y = X_r$

Decryption

- F is injective function if its second argument is fixed.
- There exists a function F^{-1} such that $F^{-1}(F(X,Y),Y) = X$. Then the decryption operation is carried out as follows:

$$\begin{aligned} X_r &\leftarrow Y \\ X_{r-1} &\leftarrow F^{-1}(X_r, K_r) \\ &\vdots \\ X_1 &\leftarrow F^{-1}(X_2, K_2) \\ X_0 &\leftarrow F^{-1}(X_1, K_1) \\ X &\leftarrow X_0 \end{aligned}$$

Substitution-Permutation Network

- plaintext: lm-bit binary string, $X = (x_1, x_2, \ldots, x_{lm})$
- we can regard X as the concatenation of m l-bit substrings: $X = X_{(1)}||\ldots||X_{(m)}$ and for $1 \leq i \leq m$, we have that $X_{(i)} = (x_{(i-1)l+1}, \ldots, x_{il})$
- S-box is a permutation $\pi_S : \{0,1\}^l \rightarrow \{0,1\}^l$
- $\pi_P : \{1, 2, \ldots, lm\} \rightarrow \{1, 2, \ldots, lm\}$
- $K_1, K_2, \ldots, K_{r+1}$ is the list of round keys derived from the secret key K
- The encryption algorithm is as follows:

Algorithm SPN

Input: $X, \pi_S, \pi_P, (K_1, \ldots, K_{r+1})$

Output: Y

1. $w^0 \leftarrow X$
2. for $i \leftarrow 1$ to $r-1$ do
3. $\quad u^i \leftarrow w^{i-1} \oplus K_i$
4. $\quad$ for $j \leftarrow 1$ to m
5. $\quad\quad$ do $v^i_{(j)} \leftarrow \pi_S(u^i_{(j)})$
6. $\quad w^i \leftarrow (v^i_{\pi_P(1)}, \ldots, v^i_{\pi_P(lm)})$
7. end do
8. $u^r \leftarrow w^{r-1} \oplus K_r$

9. for $j \leftarrow 1$ to m

10. do $v^r_{(j)} \leftarrow \pi_S(u^r_{(j)})$

11. $Y \leftarrow v^r \oplus K_{r+1}$

- Let $l = m = r = 4$ and π_S be defined as follows, where the input (i.e., z) and the output (i.e., $\pi_S(z)$ are written in hexadecimal notation,
  $(0 = (0,0,0,0), 1 = (0,0,0,1), \ldots 9 = (1,0,0,1), A = (1,0,1,0), \ldots, F = (1,1,1,1))$:

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

- Let π_P be defined as follows:

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

- **Key Scheduling Algorithm:** $K = (k_1, \ldots, k_{32})$. For $1 \leq i \leq 5$, define K_i to consist of 16 consecutive bits of K, beginning with k_{4i-3}.
- Example: Suppose the key is

$$k = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$$

- The the round keys are as follows:

$$\begin{aligned} K_1 &= 0011\ \ 1010\ \ 1001\ \ 0100 \\ K_2 &= 1010\ \ 1001\ \ 0100\ \ 1101 \\ K_3 &= 1001\ \ 0100\ \ 1101\ \ 0110 \\ K_4 &= 0100\ \ 1101\ \ 0110\ \ 0011 \\ K_5 &= 1101\ \ 0110\ \ 0011\ \ 1111 \end{aligned}$$

- Suppose that the plaintext is

$$X = 0010\ 0110\ 1011\ 0111$$

- Then the encryption of X proceeds as follows:

$$\begin{array}{rcllll}
w^0 & = & 0010 & 0110 & 1011 & 0111 \\
K_1 & = & 0011 & 1010 & 1001 & 0100 \\
u^1 & = & 0001 & 1100 & 0010 & 0011 \\
v^1 & = & 0100 & 0101 & 1101 & 0001 \\
w^1 & = & 0010 & 1110 & 0000 & 0111 \\
K_2 & = & 1010 & 1001 & 0100 & 1101 \\
u^2 & = & 1000 & 0111 & 0100 & 1010
\end{array}$$

$$
\begin{array}{lcllll}
v^2 & = & 0011 & 1000 & 0010 & 0110 \\
w^2 & = & 0100 & 0001 & 1011 & 1000 \\
K_3 & = & 1001 & 0100 & 1101 & 0110 \\
u^3 & = & 1101 & 0101 & 0110 & 1110 \\
v^3 & = & 1001 & 1111 & 1011 & 0000 \\
w^3 & = & 1110 & 0100 & 0110 & 1110 \\
K_4 & = & 0100 & 1101 & 0110 & 0011 \\
u^4 & = & 1010 & 1001 & 0000 & 1101 \\
v^4 & = & 0110 & 1010 & 1110 & 1001 \\
K_5 & = & 1101 & 0110 & 0011 & 1111 \\
Y & = & 1011 & 1100 & 1101 & 0110
\end{array}
$$

- Y is the ciphertext

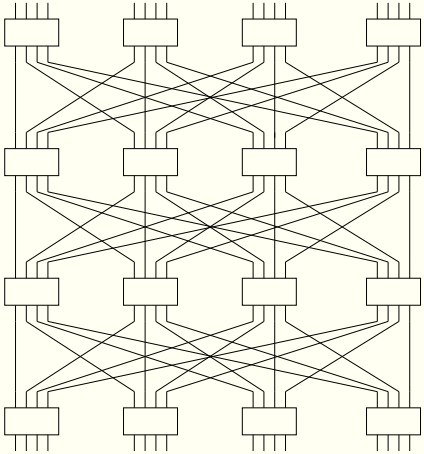

Figure 2: A Substitution-Permutation Network

Feistel Cipher

- In a Feistel cipher, each state u^i is divided into two halves of equal length, say L_i and R_i. The round function g has the following form: $g(L_{i-1}, R_{i-1}, K_i) = (L_i, R_i)$, where $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$.
- Feistel-type round function is always invertible, given the round key: $L_{i-1} = R_i \oplus f(L_i, K_i)$ and $R_{i-1} = L_i$.

Data Encryption Standard (DES)

- DES is a 16-round Feistel cipher having block length 64: it encrypts a plaintext bitstring x (of length 64) using a 56-bit key, K, obtaining a ciphertext bitstring (of length 64).
- DES is the most widely used encryption scheme, adopted in 1977 by the National Bureau of Standards, now National Institute of Standards and Technology (NIST).
- Data is encrypted in 64 bit blocks using a 56 bit key and delivering a 64 bit output (FIPS PUB 46).

- Historically DES resulted from a project first initiated by IBM in the 60's which resulted in LUCIFER, a block cipher (64 bits) which used a key size of 128 bits. The NSA got involved and the final product, DES, ended up with a 56 bit key!
- In 1994 NIST reaffirmed DES for government use for a further five years for use in areas other than "classified".

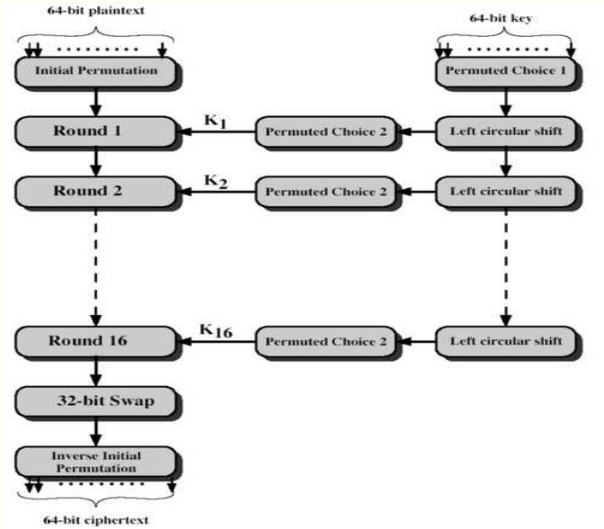

Figure 3: Block diagram of DES algorithm.

- The overall scheme for DES is shown in figure 3. Two inputs are the plaintext (64 bits) and key (56 bits). The processing of the plaintext proceeds in three phases as can be seen from the left hand side of figure:
  1. Initial permutation (**IP**) rearranging the bits to form the "permuted input".
  2. Followed by 16 iterations of the same function (substitution and permutation). The output of the last iteration consists of 64 bits which is a function of the plaintext and key. The left and right halves are swapped to produce the preoutput.

3. Finally, the preoutput is passed through a permutation ($\mathbf{IP}^{-1}$) which is simply the inverse of the initial permutation (**IP**). The output of $\mathbf{IP}^{-1}$ is the 64-bit ciphertext.

- The use of the key can be seen in the right hand portion of figure 3.
  - Initially the key is passed through a permutation function ($\mathbf{PC}_1$ - shown in table 1).
  - For each of the 16 iterations, a subkey ($\mathbf{K}_i$) is produced by a combination of a left circular shift and a permutation ($\mathbf{PC}_2$ - shown in table 1) which is the same for each iteration. However, the resulting subkey is different for each iteration because of repeated shifts.

- Detail of a single iteration: Following figure 4 and focusing on the LHS of the diagram, for the ith iteration of 16, the left and right halves are treated as separate 32 bit quantities L and R.

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Table 1: Permutation tables used in DES.

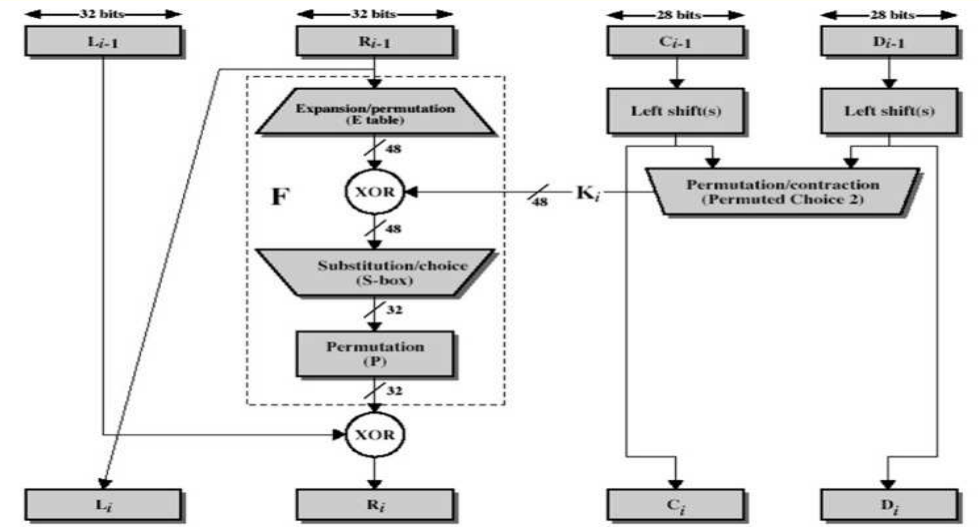

Figure 4: The details of one round of DES.

- The overall processing at each iteration is:
  - $L_i = R_{i-1}$ - i.e. the left hand output L_i of an iteration is the right hand input R_{i-1}.
  - $R_i = L_{i-1} \oplus \mathbf{F}(R_{i-1}, K_i)$.
  - $\oplus$ is the exclusive OR function - therefore the right hand output R_i is the exclusive OR of L_{i-1} and a complex function "$\mathbf{F}$" of R_{i-1} and K_i.

– The function **F** is shown in figure 5. The subkey K_i used is 48 bits and the R_{i-1} input is first expanded to 48 bits using a table that defines a permutation plus an expansion (duplication of 16 R bits as shown in table 1). The resulting 48 bits are XORed with K_i.

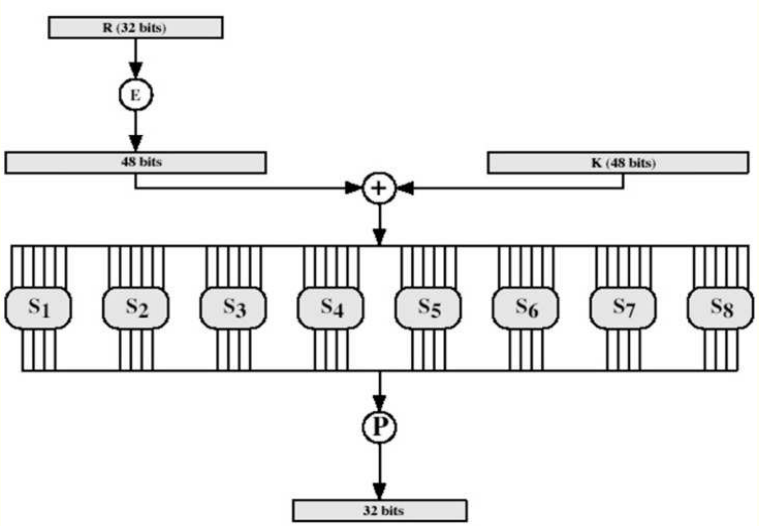

Figure 5: The complex **F** function of the DES algorithm.

- Single iteration continued:
  - This 48 bit result passes through a substitution function that produces a 32 bit output which is then permuted as shown in table 1.
  - The substitution function consists of 8 **S-boxes** each of which accepts 6 bits input and produces 4 bits of output (table 2). The first and last input bits of S_i select one of 4 rows and the four middle bits select the column. The contents of the selected cell constitute the 4 bit output.

– Key generation is shown in figure 4 where the 56 bit key used as input to the algorithm is first permuted ($\mathbf{PC}_1$ (table 3) and is then treated as two 28 bit quantities C_O and D_O).

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 2: S-box details.

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 3: DES key schedule.

- Single iteration cont:
  - At each iteration C_i and D_i are subjected to a circular left shift or rotation of 1 or 2 bits (table 3). They serve as the input to the next iteration and also are input into permuted choice 2 ($\mathbf{PC}_2$) whose output serves as $\mathbf{K}_i$ the input to $\mathbf{F}$.
- DES modes of operation: The DES algorithm is a basic building block for providing data security. To apply DES in a variety of application, five modes of operation have been defined which cover virtually all variation of use of the algorithm and these are shown in table 4

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed J bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Table 4: DES modes of operation.

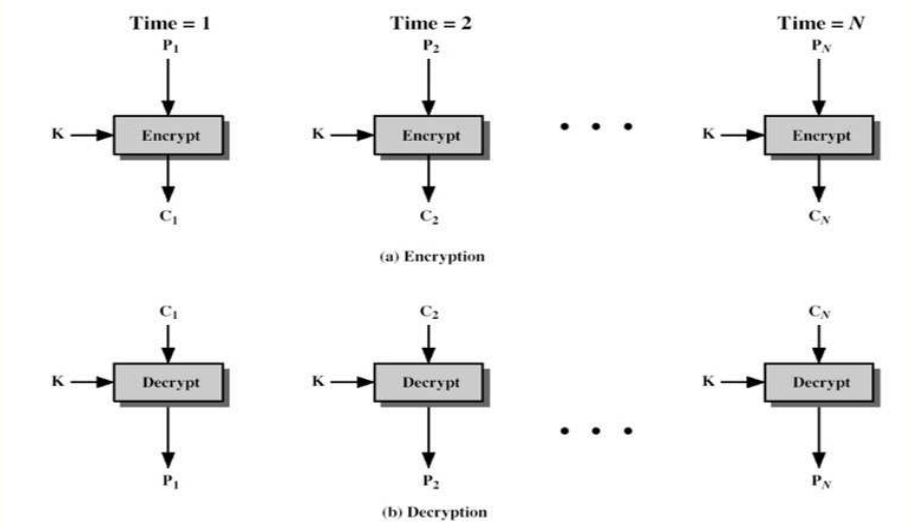

Figure 6: electronic codebook mode (ECB mode).

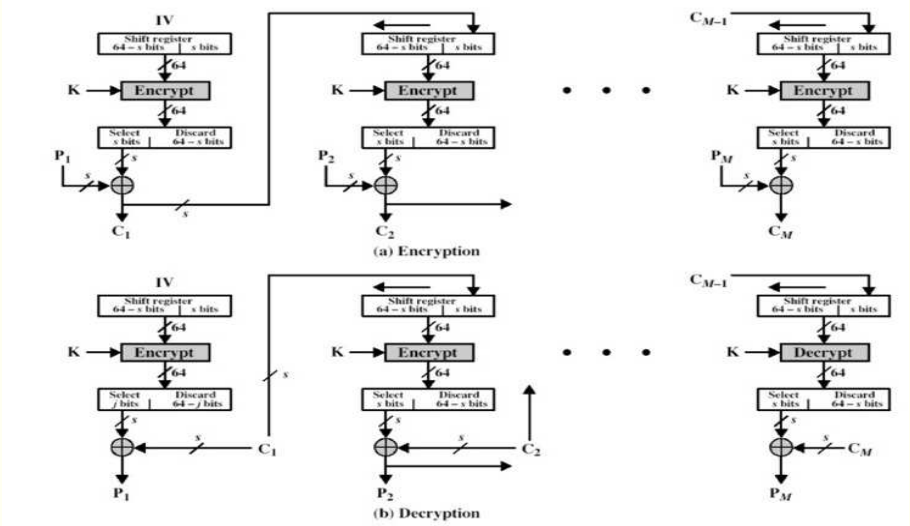

Figure 7: cipher feedback mode (CFB mode).

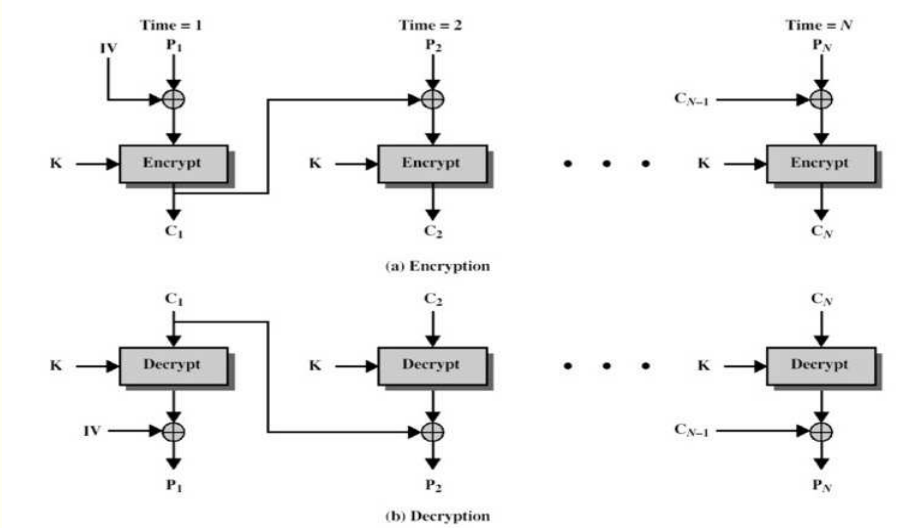

Figure 8: cipher block chaining mode (CBC mode).

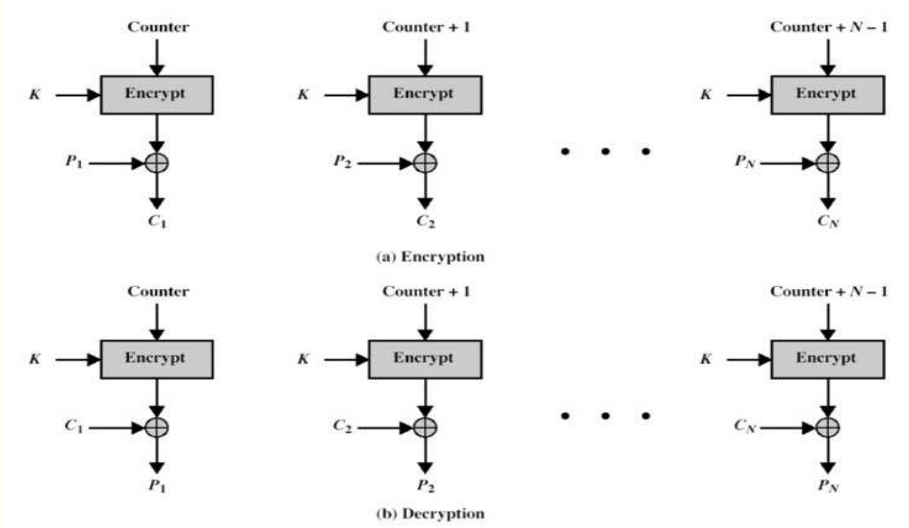

Figure 9: counter mode (CTR mode).

- The decryption process with DES is essentially the same as the encryption process and is as follows:
  - Use the ciphertext as the input to the DES algorithm but *use the keys K_i in reverse order*. That is, use K_{16} on the first iteration, K_{15} on the second until K_1 which is used on the 16th and last iteration.

- **Avalanche Effect**: A desirable property of any encryption algorithm is that a small change in either plaintext or key should produce significant changes in the ciphertext. DES exhibits a strong avalanche effect. Table 5 illustrates this.
- Concern about DES: Since its adoption as a federal standard there have been concerns about the level of security provided by DES in two areas, Key size and nature of the algorithm.

- 56 bit key length (approx. 7.2×10^{16}) on initial consideration brute-force attack seems impractical. However with a massively parallel machine of about 5000 nodes with each node capable of achieving a key search rate of 50 million keys/sec, the time taken to do a brute-force search is approximately 100 hrs which is far from excessive.
- The nature of DES algorithm: of more concern is that cryptanalysis is possible by exploiting the characteristics of DES. The focus is the eight S-boxes used in each iteration. The design criteria for the

complete algorithm has never been published and there has been speculation that the boxes were constructed in such a way that cryptanalysis is possible by an opponent who knows the weakness in the S-boxes. Although this has not been established, the US government's "clipper project" raises many questions. These are the main reasons DES is now being replaced by the AES standard which we will look at later on.

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Table 5: Avalanche effect - a small change in the plaintext produces a significant change in the ciphertext.

Exhaustive Search on DES

- Diffie-Hellman (1977):
  - Special purpose hardware;
  - Estimated time 12-hours at a cost of USD 20M.
- Wiener (1993):
  - Special purpose hardware;
  - Estimated time 3.5 hours at a cost of USD 1 M.

Exhaustive Search on DES

- Goldberg-Wagner (1996):
  - FPGA based hardware;
  - Estimated time one year at a cost of USD 45,000.
- Electronic Frontier Foundation (EFF) (1998): DES Cracker
  - Special purpose hardware;
  - Estimated time 3 days at a cost of USD 200,000.

- Only perfectly secure algorithm we have is not very practical for use in the field.
- Necessary to turn back to some of the other algorithms and see what can be done to improve security.
- Because DES was standard it was desired to keep it in the picture and as a result the algorithm **Triple DES** was realised.
- Triple DES is simply applying the DES algorithm three times using two different keys (see figure 10):

$$C = E_{K_1}[D_{K_2}[E_{K_1}[P]]] \tag{1}$$

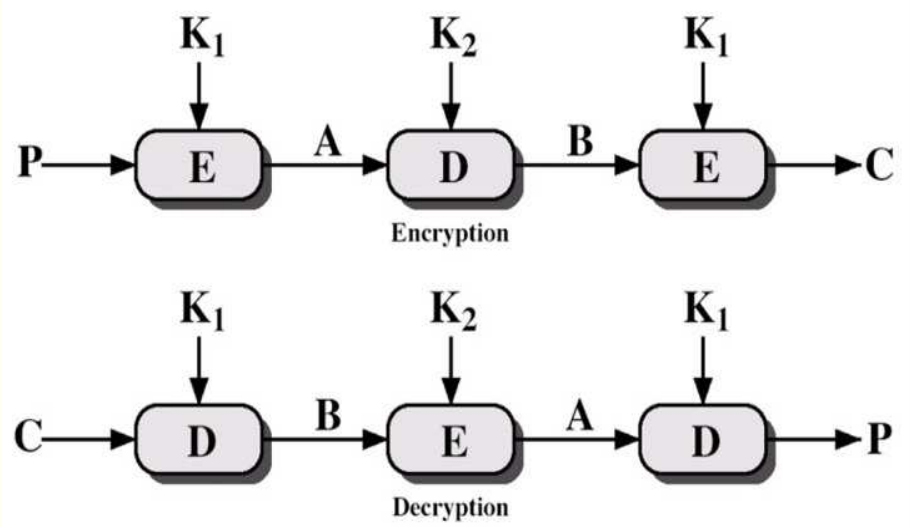

Figure 10: Triple DES.

- The order of encryptions and decryptions determines name EDE or Encrypt-Decrypt-Encrypt.
- Decryption is performed on the second iteration is simply for backward compatibility and offers no extra security to the algorithm.
- This can be seen from:

$$C = E_{K_1}[D_{K_1}[E_{K_1}[P]]] = E_{K_1}[P]. \tag{2}$$

Stream Ciphers

- Plaintext: binary string.
- Key-stream: a pseudorandom binary string.
- Ciphertext: bit-wise XOR (addition modulo 2) of plaintext and key-stream.
- Decryption: bit-wise XOR of ciphertext and key-stream.
- **Example:**

$$
\begin{aligned}
P &: \ 10001101010111011011 \\
K &: \ 01001010110100110110 \\
C &: \ 11001111100011011010
\end{aligned}
$$

Shannon's Notion of Perfect Secrecy

- $\Pr(x \mid y) = \Pr(x)$ for all x and for all y where x is a plaintext and y is a ciphertext.
- The basic strength of stream-cipher lies in how "random" the key-stream is.
- If k_i is a true random sequence, then the cipher is called an one-time pad.
- One-time pad possesses *perfect secrecy*. However, one-time pad is impractical.
- Main objective of a stream cipher construction is to get k_i as random as possible.

Illustration

One bit encryption, $C = P \oplus K$.

$\Pr(K = 0) = \Pr(K = 1) = \frac{1}{2}$.

Let $\Pr(P = 0) = 0.6,\ \Pr(P = 1) = 0.4$.

$$\Pr(P = 0 \mid C = 1) = \frac{\Pr(P{=}0,\ C{=}1)}{\Pr(C{=}1)}$$

$$= \frac{\Pr(P{=}0,\ C{=}1)}{\Pr(P{=}0,\ C{=}1){+}\Pr(P{=}1,\ C{=}1)}$$

$$= \frac{\Pr(C{=}1|P{=}0).\Pr(P{=}0)}{\Pr(C{=}1|P{=}0).\Pr(P{=}0){+}\Pr(C{=}1|P{=}1).\Pr(P{=}1)}$$

$$= \frac{\Pr(K{=}1).\Pr(P{=}0)}{\Pr(K{=}1).\Pr(P{=}0){+}\Pr(K{=}0).\Pr(P{=}1)} = \frac{\frac{1}{2}\times 0.6}{\frac{1}{2}\times 0.6{+}\frac{1}{2}\times 0.4} = 0.6.$$

Randomness Measurements

- Randomness of sequence: unpredictable property of sequence.
- Aim is to measure randomness of the sequence generated by a deterministic method called a generator.
- The test is performed by taking a sample output sequence and subjecting it to various statistical tests to determine whether the sequence possess' certain kinds of attributes, a truly random sequence would be likely to exhibit.

- This is the reason the sequence is called **pseudo-random sequence** instead of random sequence and the generator is called **pseudo-random sequence generator (PSG)**.
- The sequence $s = s_0, s_1, s_2, \ldots$ is said to be *periodic* if there is some positive integer N such that $s_{i+N} = s_i$ and smallest N is called the *period* of sequence.
- Golomb's Randomness Postulates is one of the initial attempts to establish some necessary conditions for a periodic pseudorandom sequence to look random.

Golomb's Randomness Postulates

R-1 In every period, the number of 1's differs from the number of 0's by at most 1. Thus $|\Sigma_{i=0}^{N-1}(-1)^{s_i}| \leq 1$.

R-2 In every period, half the runs have length 1, $\frac{1}{4}$th have length 2, $\frac{1}{8}$th have length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are (almost) equally many runs of 0's and of 1's.

R-3 The auto-correlation function $C(\tau) = \Sigma_{i=0}^{N-1}(-1)^{s_i+s_{i+\tau}}$ is two-valued. Explicitly

$$C(\tau) = \begin{cases} N & if\ \tau \equiv 0(mod N) \\ T & if\ \tau \not\equiv 0(mod N) \end{cases}$$

where T is a constant.

Example: Consider the periodic sequence s of period 15 with cycle $s^{15} = 011001000111101$.

R-1: There are seven 0's and eight 1's.

R-2: Total runs is 8. 4 runs of length 1 (2 for each 0's and 1's), 2 runs of length 2 (1 for each 0's and 1's), 1 run of 0's of length 3 and 1 run of 1's of length 4.

R-3: The function $C(\tau)$ takes only two values: $C(0) = 15$ and $C(\tau) = -1$ for $1 \leq \tau \leq 14$.

Five Basic Tests

- **Frequency test** (monobit test): To test whether the number of 0's and 1's in sequence s are approximately the same, as would be expected for a random sequence.
- **Serial test** (two-bit test): To determine whether the number of 00, 01, 10 and 11 as subsequences of s are approximately the same, as would be expected for a random sequence.
- **Runs test:** To determine whether the number of runs of various lengths in the sequence satisfy the R-2, as expected for a random sequence.

- **Poker test:** Let m be a +ve integer. Divide the sequence into $\lfloor \frac{n}{m} \rfloor$ non-overlapping parts of length m. To test whether the number of each sequence of length m are approx. the same, as would be expected for a random sequence.
- **Auto-correlation test:** To check whether correlation between the sequence and shifted version of it is approximately 0 when the number of shifts is not divisible by the period as expected for a random sequence.

- Random numbers play an important role in this area.
- Two criteria are used to validate the randomness of a sequence of numbers:
  1. **Uniform distribution** - the frequency of occurrence of each of the nos. should be approximately the same.
  2. **Independence** - no one value in the sequence can be inferred from the others.

- Many tests to determine whether a sequence has uniform distribution. No test to prove independence.
- No. of tests can be applied to demonstrate that a sequence doesn't exhibit independence.
- Sources of true random numbers are hard to come by.
- Deterministic algorithmic techniques are used instead and if the algorithm is good, the resulting sequence will pass many tests of randomness.
- These are called **PseudoRandom Number Generators (PRNG)**.

- The most widely used technique is **linear congruential method**. The algorithm is parameterised as follows:

m	the modulus	$m > 0$
a	the multiplier	$0 < a < m$
c	the increment	$0 \leq c < m$
X_0	the starting value, or seed	$0 \leq X_0 < m$

- The sequence of random nos. X_n is obtained via the following iterative equation:

$$X_{n+1} = (aX_n + c) \bmod m \tag{3}$$

- If m,a,c and X_0 are integers then the technique will produce a sequence of integers with each integer in the range $0 \leq X_n < m$.
- The selection of a,c and m is critical in developing a good generator.

- Consider the following:
  - For $a = c = 1$ the sequence produced is no use.
  - Consider $a = 7, c = 0, m = 32$ and $X_0 = 1$. The resultant sequence is also clearly unsatisfactory $\{7, 17, 23, 1, 7, \ldots\}$.
  - Of 32 possible values only 4 are used (i.e. the sequence has period 4). Changing $a = 5$, then the sequence is $\{5, 25, 29, 17, 21, 9, 13, 1, \ldots\}$ with period 8.
  - m should be large so that there is potential for long sequences, usually m is nearly equal to the max. nonnegative integer for a given computer e.g. $2^{31} - 1$.

- Three criteria are used to assess random no. generators as follows:
  - T_1 : The function should be a full-period generating function (i.e. should generate all nos. between 0 and m before repeating).
  - T_2 : The generated sequence should appear random. The sequence is not random at all but there is a wide variety of tests to assess the degree of randomness exhibited.
  - T_3 : The function should implement efficiently with 32-bit arithmetic.

- With appropriate values of a, c and m, these criteria may be achieved.
- If m is prime and c $= 0$, then for certain values of a the period of the generated sequence is $m - 1$, with only the value 0 missing. For 32 bit arithmetic, a convenient large prime of m is $2^{31} - 1$ and the function is as follows:

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

- Of the more that 2 billion choices for a only a handful of multipliers pass all three tests. One of these is $a = 7^5 = 16,807$, which was originally designed for use in the IBM360 family of computers and tested extensively.

- Strength of method is that resultant sequence can be *indistinguishable* from a sequence drawn randomly (but without replacement) from the set $1, 2, 3, 4, \ldots, m-1$.
- Nothing random about the algorithm, apart from X_0. Remaining numbers in the sequence follow deterministically.
- If opponent knows the algorithm is being used, he can easily discover the sequence.
- Is possible to solve for a, c, m.

- To make the sequence less reproducible, the sequence could be restarted every N nos.
- This can be achieved using the current clock value $(\mathrm{mod}(m))$ as new seed after N numbers (starting the new sequence).
- Another method could be to add the current value of the clock $(\mathrm{mod}(m))$ to the random numbers produced.
- Cryptographically generated random nos.: It makes sense to take advantage of the encryption logic available to produce random nos. Some examples follows:

- **Cyclic Encryption**: fig. 11 illustrates a method of generating a session key from a master key. A counter with period N provides input to the encryption logic. If 56 bit DES keys are to be produced, then a counter with a period of 2^{56} may be used. After each key is produced the counter is incremented, thus the pseudorandom nos. generated by the scheme cycle through a full period. To further strengthen the algorithm the counter could be replaced by a full period PRNG.

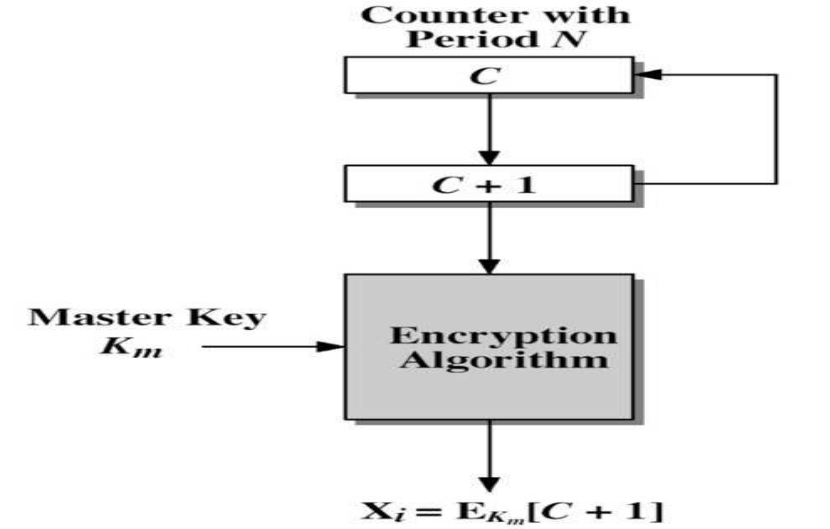

Figure 11: Pseudorandom number generation from a counter.

ANSI X9.17: Pseudorandom no. generator.

- Considered one of the strongest (cryptographically speaking).
- U.S. Federal Information Processing Standard (FIPS) approved method
- Makes use of Triple DES to produce random numbers.

- Figure 12 shows the scheme. The algorithm makes use of triple DES and has the following:
  - Input: Two pseudorandom inputs drive the algorithm. One is a 64 bit representation of the current date and time updated on each no. generation. The other is a 64 bit seed value initialized to some arbitrary value and updated during the generation process.
  - Keys: The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56 bit keys, which must be kept secret.
  - Output: These are a 64 bit pseudorandom no. (R_i) and a 64 bit seed value (V_{i+1}).

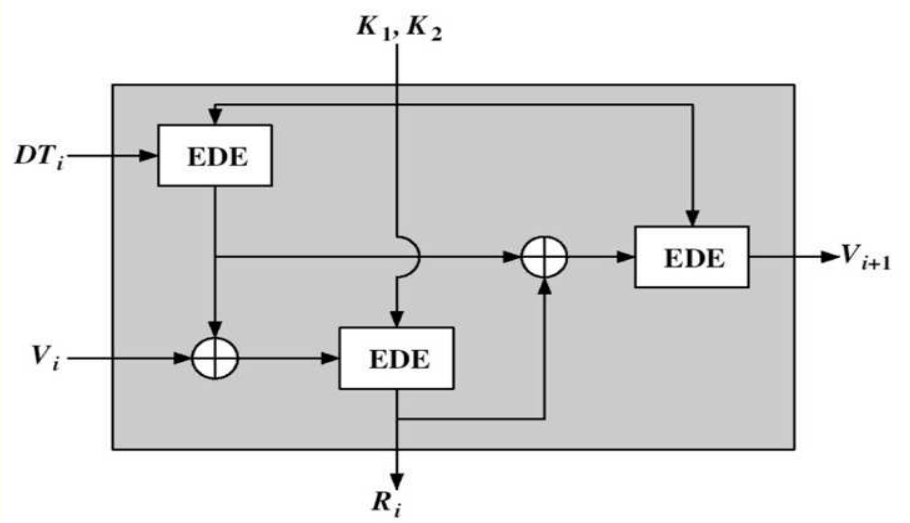

Figure 12: ANSI X9.17 Scheme.

- The strength of the PRNG of figure 12 can be attributed to the following:
  - There are a total of nine DES encryptions and a 112 bit key (effectively).
  - The scheme takes two pseudorandom inputs to begin with, both of which are not revealed (although it might be possible to work out DT_i).
- As a result the amount of information needed by and attacker is formidable.
- Even if the attacker learns R_i, it would be impossible to deduce v_{i+1} due to extra EDE encryption.